Design of Self-cycling QCA Multiplier

Mingwei Jin, Peiqiao Wu, Xingguo Xiong
Department of Electrical Engineering, University of Bridgeport, 221 University Avenue, Bridgeport, CT 06604

Abstract

Quantum cellular automaton technology (QCA) is a newly-developed technology for next-generation nanoelectronics. It results in high device density, ultra-fast speed and almost zero power dissipation. Multiplier is an important unit in microprocessor design. Traditional N-bit array multiplier consists of N*N adder units and results in large circuit area. In this poster, a novel area-efficient self-cycling architecture for N-bit multiplier is proposed. It ingeniously utilizes the inherent pipeline feature of QCA clock zones to convert the array multiplication into a serial-in serial-out process. In this way, the N-bit multiplier can be implemented with a single (instead of N*N) adder unit, which leads to significant area saving. The proposed self-cycling architecture can be extended to other N-bit QCA circuits as well. It is especially suitable for area-critical QCA circuit design where serial output can be accepted.

Keywords: Quantum-dot Cellular Automata (QCA), Multiplier, Self-cycling Architecture, QCADesigner.

Introduction

Quantum Cellular Automata (QCA) is a newly-developed technology for next-generation nanoelectronics. It utilizes charge polarity instead of voltage to represent digital “0” and “1” states. The working principle is based on the electron distribution among quantum dots inside the QCA cells. In this poster, the design and simulation of a self-cycling multiplier structure are reported.

Self-cycling QCA Multiplier

1). Carry-Self-cycling multiplier design

As we observe the structure of NXN multiplier, it contains many identical multiplication-addition units. Each adder unit takes the carryout from previous unit. In this way, we may combine multiple units into a single unit, but use pipeline structure to store the outputs in series. As every unit will pass the carryout to next unit, why not let them become input again? In this way we can use one unit to finish what four slice-bit units do (in 4x4 multiplier), as shown in Figure 1. This theory is based on different clock zones can save different signal. Such as when combining four units into one, the first output S_{a0b0} (SUM of a0b0) will be saved in first clock zone, then S_{a1b0}; then S_{a2b0}; then S_{a3b0}; finally C_{a3b0} (Carry a3b0). Every output comes as the clock zone sequence.

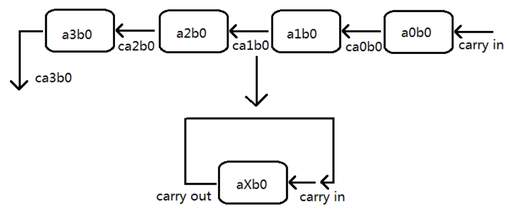

Figure 1. Combining four stages into one

By doing this a 4x4 bit QCA pipelined multiplier can become four signal units in series. As shown in Figure 2, 16 bit-slice units have been combined to 4 units. The output sequence is presented as clock zone number increasing.

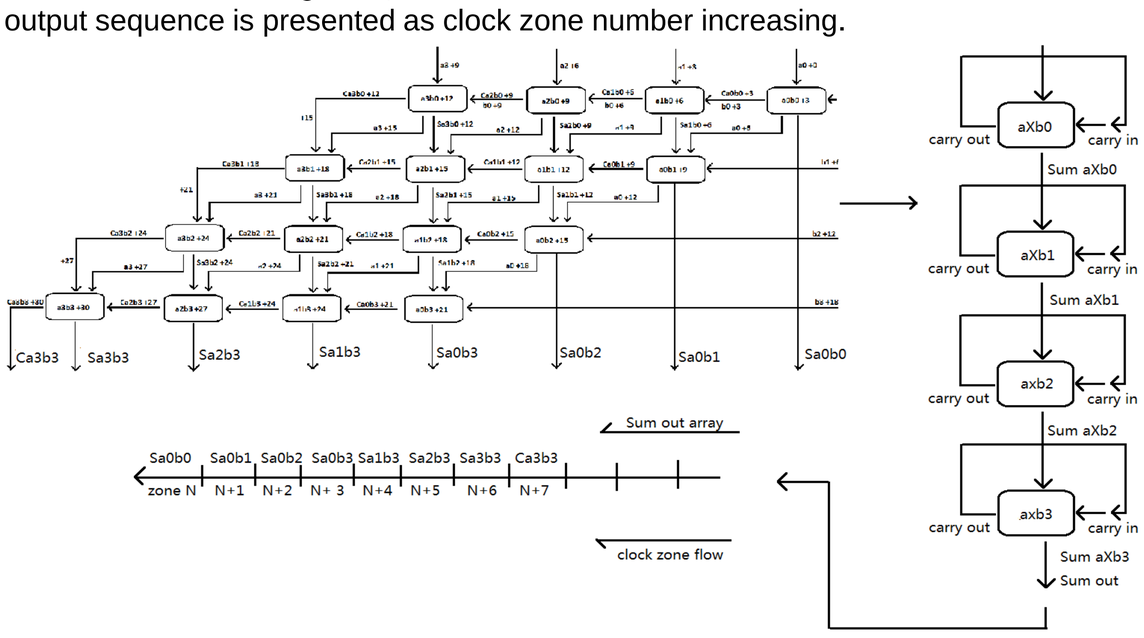

Figure 2. combine 4x4 pipelined multiplier into 4 single units in series

In Figure 3, we can see the Carry-self-cycling unit design. It is very similar to slice-bit unit but simpler. The Carry-self-cycling part is in the center. Because self-cycling part will have one clock zone delay passing to carryout (reaching the final majority gate), other wire (Cin3) should have one clock zone delay to make sure the inputs of all final majority gates arrive at the same time.

3). Self-cycling multiplier design

Since Carry out can become self-cycling, why not Sum out? As shown in Figure 4, if carry-self-cycling Sum output is used as its input, what will happen? 4 units will finally be combined into one; it will save significant area and make the design much more compact. The output sequences are showing as the output lane in Figure 4.

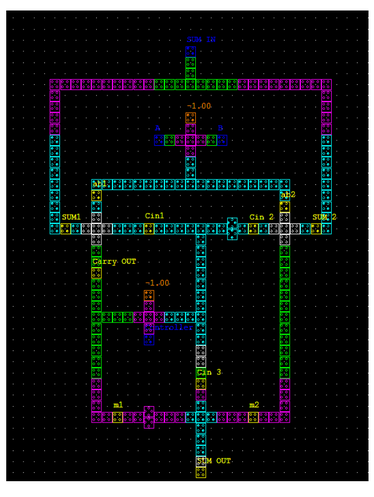

Figure 3. Carry-self-cycling unit

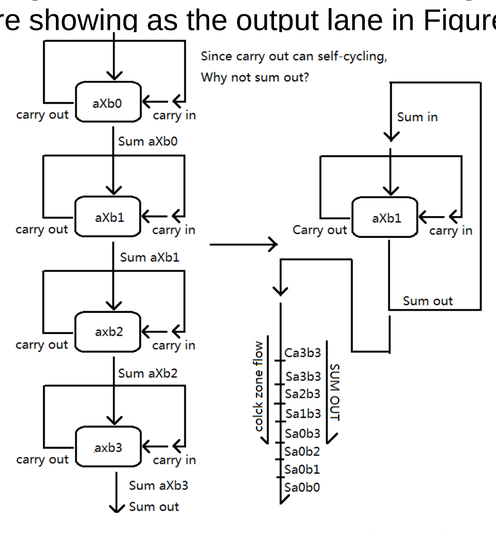

Figure 4. combined 4 carry self-cycling units to one

Figure 5 is a very important logic explanation of this self-cycling unit. Every logic cycle (or in other words, clock cycle), this unit will do one calculation showing in the rectangles of Figure 5, from right to left, then from top to bottom. Some Dashed-line rectangles mean that there should be add an imaginary unit to let sum out pass to next or to let carry out transform to sum out(with input a=0, b=0, sum in= 0) in order to let it become input for latter unit.

4). Simulation

For easier adjust the clock zone during design this self-cycling unit, I took 1111×1111 as an example test pattern because it results in many carries, which is good to show how the clock zones should be adjusted.

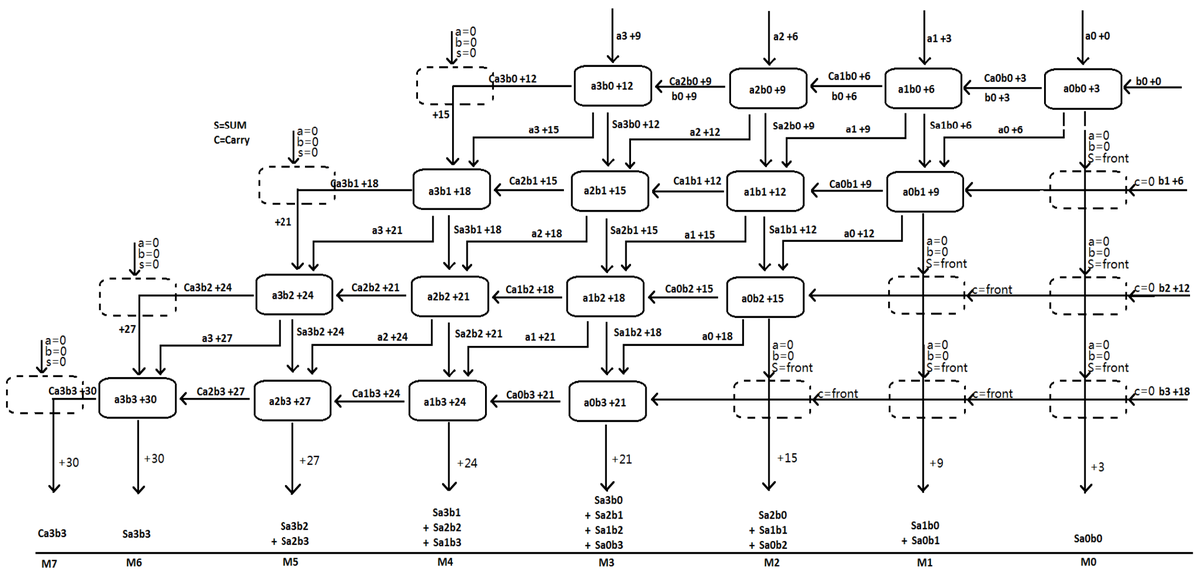

Figure 5. Logic explanation of 4X4 self-cycling multiplier

Figure 6 is the structure of this self-cycling unit. It can make Carryout and Sum out self-cycling. It is very convenient to design by using very low area wasting and cells number. Which also means simulation can be finished very soon. There is another controller-controller S, it is used to control the sum output cycle. Same to the controller C, when it is set to 0, output signal of that clock cycle will be set to zero.

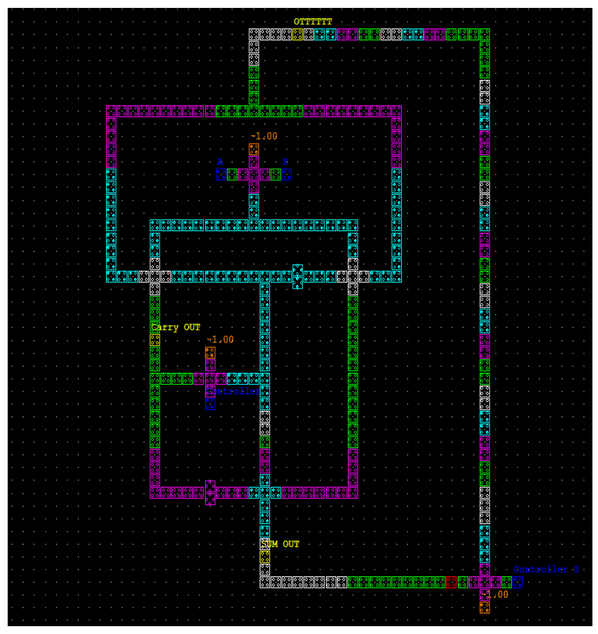

Figure 6. Combining four carry self-cycling units to one

	A	B	C	D	E	F	G	H	I
1	unit	a	b	ab	Cin	Sin	CarryOut	SumOut	ControC
2	a0b0	1	1	1	0	0	0	1	0
3	a1b0	1	1	1	0	0	0	1	1
4	a2b0	1	1	1	0	0	0	1	1
5	a3b0	1	1	1	0	0	0	1	1
6	Ca3b0	0	0	0	0	0	0	0	1
7	a0b0	0	0	0	0	1	0	1	0
8	a0b1	1	1	1	0	1	1	0	1
9	a1b1	1	1	1	1	1	1	1	1
10	a2b1	1	1	1	1	1	1	1	1
11	a3b1	1	1	1	1	0	1	0	1
12	Ca3b1	0	0	0	1	0	0	1	1
13	a0b0	0	0	0	0	1	0	1	0
14	a0b1	0	0	0	0	0	0	0	1
15	a0b2	1	1	1	0	1	1	0	1
16	a1b2	1	1	1	1	1	1	1	1
17	a2b2	1	1	1	1	0	1	0	1
18	a3b2	1	1	1	1	1	1	1	1
19	Ca3b2	0	0	0	1	0	0	1	1
20	a0b0	0	0	0	0	1	0	1	0
21	a0b1	0	0	0	0	0	0	0	1
22	a0b2	0	0	0	0	0	0	0	1
23	a0b3	1	1	1	0	1	1	0	1
24	a1b3	1	1	1	1	0	1	0	1
25	a2b3	1	1	1	1	1	1	1	1
26	a3b3	1	1	1	1	1	1	1	1
27	Ca3b3	0	0	0	1	0	0	1	1

Figure 7. Predicted truth table of 4X4 multiplier

The output lane totally has 7 clock zone delay. It was set to contain the whole output information. For instance, for 4×4 bit multiplier, output sequence will be Sa0b0, Sa0b1, Sa0b2, Sa0b3, Sa1b3, Sa2b3, Sa3b3, Ca3b3. Two signals are running in this unit (this unit runs with two clock zone delays), others will be stored in output line. Therefore, output line’s clock zone should be enough to store these signals.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1	ClockZone	0~8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
2	a	0	0	a0	a1	a2	a3	0					0	a0	a1	a2	a3	0				0	0	
3	b	0	0	b0	b0	b0	b0	0					0	b1	b1	b1	b1	0				0	0	
4	Contro C	0	0	1	1	1	1	1					0	1	1	1	1	1				0	0	
5	Sin	0	0	0	0	0	0	0					Sa0b0	Sa0b1	Sa0b2	Sa0b3	Ca0b3	0						
6	Sout					Sa0b0	Sa0b1	Sa0b2	Sa0b3	Ca0b3	0				Sa0b0	Sa0b1	Sa1b1	Sa2b1	Sa3b1	Ca3b1	0			
7	Sout(Num)					1	1	1	1	0					1	0	1	1	0	1				
8																								
9	ClockZone		28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
10	a		0	0	a0	a1	a2	a3	0			0	0	0	a0	a1	a2	a3	0					
11	b		0	0	b2	b2	b2	b2	0			0	0	0	b3	b3	b3	b3	0					
12	Contro C		0	1	1	1	1	1	1			0	1	1	1	1	1	1	1					
13	Sin		Sa0b0	Sa0b1	Sa1b1	Sa2b1	Sa3b1	Ca3b1	0			Sa0b0	Sa0b1	Sa0b2	Sa1b2	Sa2b2	Sa3b2	Ca3b2	0					
14	Sout				Sa0b0	Sa0b1	Sa0b2	Sa1b2	Sa2b2	Sa3b2	Ca3b2	0		Sa0b0	Sa1b0	Sa2b0	Sa3b0	Sa3b1	Sa3b2	Sa3b3	Ca3b3			
15	Sout(Num)				1	0	0	1	0	1	1			1	0	0	0	0	1	1	1			

Figure 8. Sequences of signals for input pattern 1111X1111

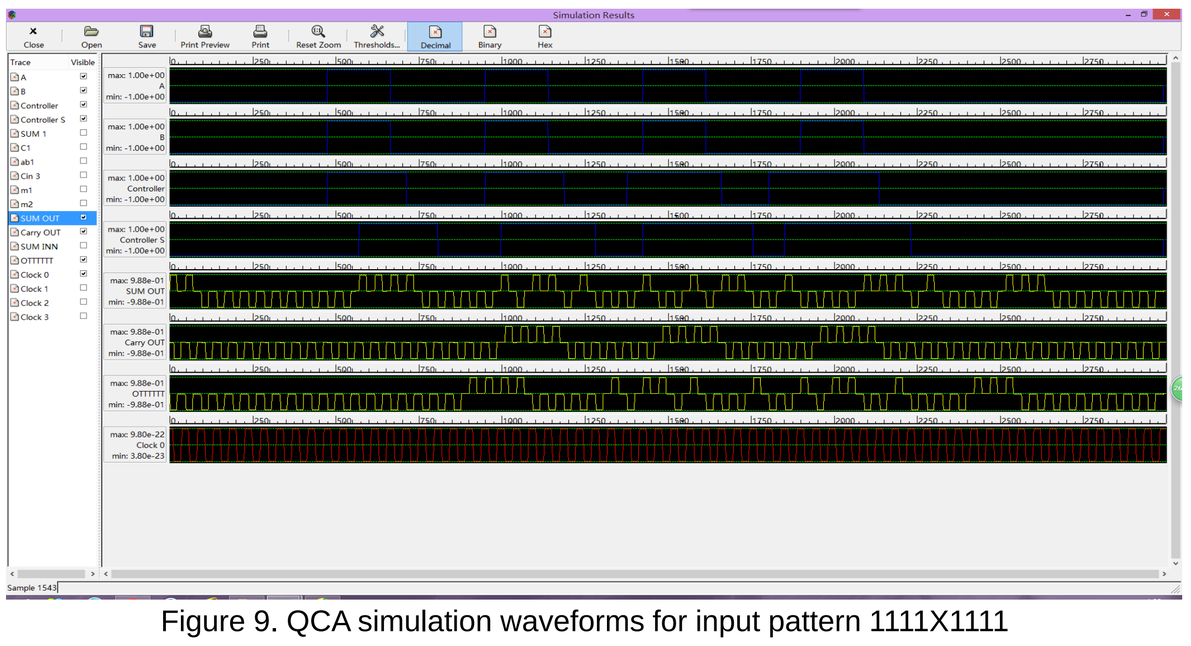

Figure 9. QCA simulation waveforms for input pattern 1111X1111

As Figure 9 shown, output pattern perfectly matches predicted signal values in Figure 8, lane- Sout(Num)). It verifies the correct function of this self-cycling unit. Furthermore, for more bit self-cycling multiplier, just change the input table and output delay cycles’ number, It will work correctly as expected. It is very simple for N bit design.

6). Comparison between pipelined and self-cycling multiplier

Table 1. comparison between pipelined and self-cycling multiplier

QCA Multiplier type(e.g.4x4)	cells	Area(μm²)	Simulation speed	Simulation Parameter setting	Use as any bit	Result delay	Several input process
Pipelined multiplier	6396	28.125	~ 20min	precision	no	30 clock cycles	yes
Self-cycling multiplier	450	0.6475	~ 3 min	As normal	yes	46 clock cycles	no

When compared to pipelined QCA multiplier, it is a trade-off, self-cycling multipliers are using more delay to trade for area.

References

[1] Ismo Hanninen and Jarmo Takala "Pipelined Array Multiplier Based on Quantum-Dot Cellular Automata“, 18th European Conference on Circuit Theory and Design, 2007, pp. 938-941.